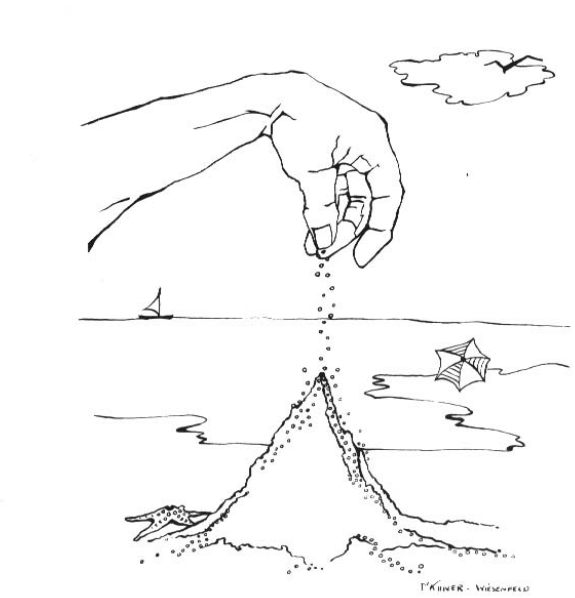
Tarea 1: Pilas de Arena

Profesores: Nelson Baloian
Patricio Poblete
Auxiliares: Manuel Cáceres
Michel Llorens
Sergio Peñafiel

Fecha de Entrega: 7 de Abril 23:59hrs



1 Introducción

Muchas veces la descripción de problemas físicos puede ser complicada y los modelos analíticos que estos desprenden pueden llegar a ser imposibles de resolver usando las herramientas de cálculo convencionales.

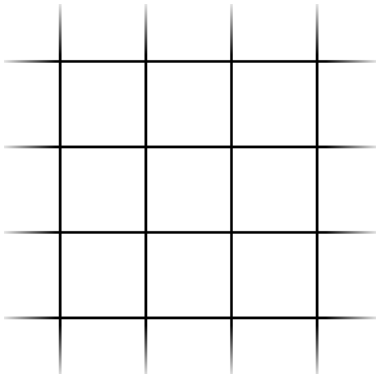
Para este tipo de problema es común recurrir a la ayuda de una modelación computacional junto con un algoritmo que describa o aproxime las leyes físicas. Los resultados, si bien son aproximaciones, pueden dar un buen indicio del comportamiento real del problema.

En esta tarea se estudiará el fenómeno de granos de arena apilados en el espacio, a estos se les aplicará una regla de estabilidad por gravedad y se visualizarán las estructuras resultantes.

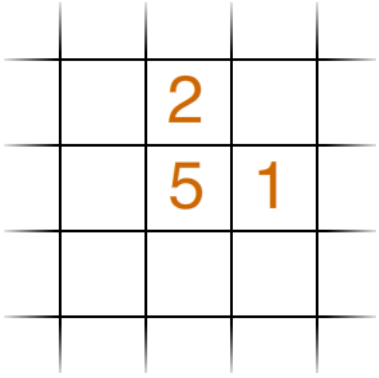
2 Explicación

2.1 Pilas de Arena

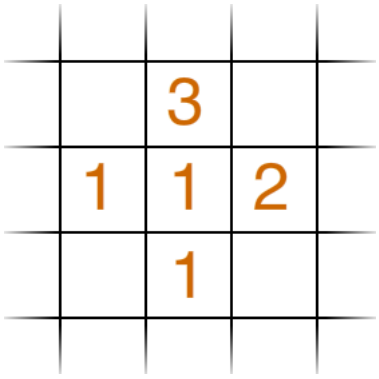
Para comenzar el modelamiento se tiene una superficie plana potencialmente infinita donde se depositarán los granos de arena. La superficie se discretizará en celdas cuadradas de igual tamaño y forma generando una malla como se muestra en la figura siguiente.



En cada celda se tendrá información de cuantos granos de arena apilados hay en esa celda. Así por ejemplo la siguiente configuración indica que en la celda central hay una pila de 5 granos en la superior hay 2, y a la derecha 1. El resto no contiene granos de arena.



Dado que las celdas de la grilla representan regiones del espacio muy pequeñas, es posible pensar que no podemos tener una cantidad muy grande de granos apilados en una misma celda dado que este sistema se volvería inestable. Para este modelo supondremos que cuando existen 4 o más granos de arena en una celda, entonces esta se desborda dejando un grano en cada una de las celdas vecinas (arriba, abajo, derecha e izquierda). De esta forma el ejemplo anterior sería inestable y al aplicar la regla quedaría en la configuración que se muestra en la figura.



Bajo este principio, un problema surge al tener más de una celda inestable pues no se especifica a cuál debe aplicarse la regla primero, sin embargo es posible demostrar que el dominio definido junto con la regla anterior generan un grupo **abeliano** lo cual nos indica que si existen 2 celdas A y B inestables, aplicar la regla sobre A y luego sobre B genera el mismo resultado que haberla aplicado sobre B y luego A. Luego el orden en que se apliquen las reglas no es relevante.

2.2 Visualización

Para esta tarea dispondrá de la clase `Ventana` que le ayudará en la visualización. La definición es la siguiente.

`Ventana(int tamaño, String titulo)`	Constructor, crea y muestra una nueva ventana vacía cuadrada del tamaño y titulo especificados como parámetros.
`mostrarMatriz(int[][] mat)`	Dibuja en la ventana la matriz indicada como cuadros de colores. Esta debe contener valores entre 0 y 3 (ambos incluidos).

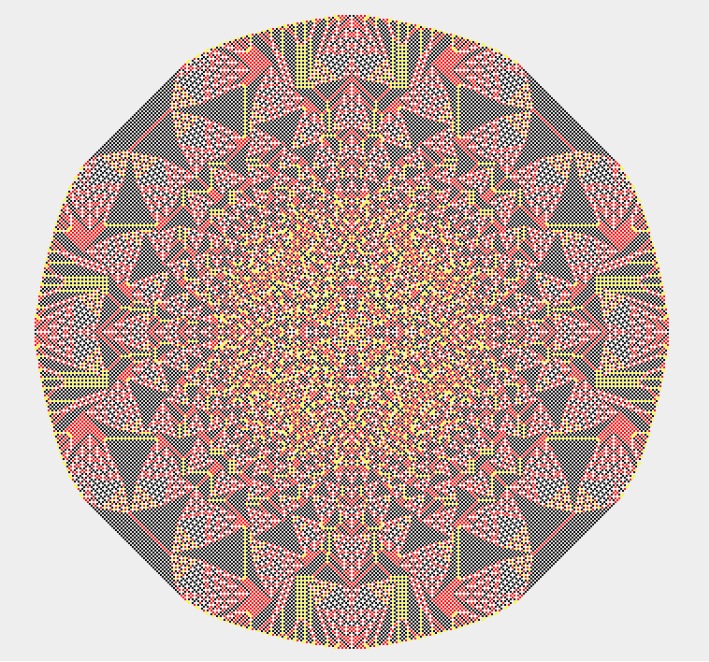
3 Implementación

En esta tarea usted deberá entregar un archivo `PilaArena.java` que hará una simulación de las pilas de arenas descritas anteriormente, en particular simularemos las estructuras que se generan al estabilizar una pila de granos muy alta en una única celda.

Para esto cree un programa que realice lo siguiente:

1. Pida al usuario un entero N que indica la cantidad de granos que se depositarán en el centro.
2. Cree una matriz de enteros de un tamaño apropiado. Esta describirá los granos de arena en cada celda. Por esto debe inicializarla con 0 excepto en la celda central cuyo valor debe ser N.
3. Aplique la regla de estabilidad, para esto primero debe ver celda a celda si es necesario aplicar la regla de desborde o no, repita este procedimiento hasta que el sistema completo sea estable.
4. Cree una ventana usando la clase provista y muestre la matriz resultante en pantalla.

En la imagen siguiente se muestra el resultado esperado usando N = 100000. El cual tomó cerca de 30s en ejecutarse.



Caso de borde

Dado que en el modelo original la matriz es potencialmente infinita no hay problema con el desborde pues siempre existen vecinos a una celda. En la implementación anterior, la matriz es finita y los bordes no tienen todos sus vecinos, por lo tanto debe considerar este caso en su solución y explicarlo en el informe.

Algunas soluciones válidas son:

- Dado el valor de N encontrar una cota superior para el tamaño de la matriz, justificándolo en el informe.
- Suponer que el borde es un sumidero y por lo tanto siempre tiene valor 0 aun cuando le lleguen granos.
- Si se necesita más espacio en la matriz, copiarla a una de tamaño más grande y seguir iterando.

No es aceptable que el programa no responda si ocurre este caso.

4 Reglas

- Esta tarea debe ser resuelta en Java.
- Es obligatorio la entrega de un informe en formato pdf junto con su tarea (Ver siguiente sección).
- Esta tarea es de carácter individual, cualquier caso de copia se evaluará con la nota mínima.
- No olvide subir a U-cursos **todos** los archivos necesarios para que su tarea funcione correctamente.
- Debe subir los archivos de código fuente (*.java). Los archivos compilados (*.class) no serán evaluados.
- Cualquier duda respecto a la tarea puede ser consultada usando el foro del curso.
- No se aceptan atrasos.

5 Informe

El informe debe describir el trabajo realizado, el código fuente desarrollado, los resultados obtenidos y las conclusiones o interpretaciones de estos. Principalmente sea conciso, describiendo cada uno de los puntos que a continuación se indican.

- **Portada:** Indicando número de la tarea, fecha, autor, email, código del curso.
- **Introducción:** Descripción breve del problema y su solución.
- **Análisis del problema:** Exponga en detalle el problema, los supuestos que pretende ocupar, casos de borde y brevemente la metodología usada para resolverlo.
- **Solución del problema:**
  - Algoritmos de solución, incluyendo toda la información y figuras que considere necesarias.
  - Partes relevantes del código fuente
  - Ejemplos de entradas y salidas escogidos por usted.
- **Modo de uso:** explicando brevemente cualquier dato extra necesario para la compilación y ejecución de su programa.
- **Resultados y análisis:** Todo el análisis de los resultados, los gráficos, imágenes y la discusión requerida.